Comput. & Ops Res., Vol. 5, pp. 11–30. Pergamon Press, 1978. Printed in Great Britain

EVALUATING DISPATCHING CONSEQUENCES OF AUTOMATIC VEHICLE LOCATION IN EMERGENCY SERVICES

RICHARD C. LARSON* and EVELYN A. FRANCK†
Operations Research Center, Massachusetts Institute of Technology

Scope and purpose—This paper develops a computer-implemented analytical model to serve as an evaluation tool for a new technology: automatic vehicle location (AVL) systems. AVL systems are currently being developed for use in a variety of urban services, including police, transportation, and postal. AVL cost is projected to range from $500 to over $10,000 per vehicle. Focusing primarily on police, the model builds on the "hypercube queuing model" reported earlier in this journal. The revised model allows for "AVL" dispatching of police patrol units, meaning that the *closest available* unit is always dispatched to the scene of the indident. The model allows a detailed depiction of a city's geography and travel time characteristics as well as rather flexible procedures for deploying patrol units. Thus, using the model, a potential consumer, manufacturer or evaluator of an AVL system can estimate the effect of AVL on response times, inter-area dispatch frequencies, workloads, dispatch error probabilities, and other measures of system performance.

Abstract—Automatic vehicle location (AVL) systems present to the dispatcher of emergency response units (e.g., police cars, ambulances) the estimated real time locations of units within his service area. Building on a recently developed hypercube queuing model, this paper presents a Markov process model for computing the operating characteristics of the radio-dispatched fleet operating under a policy that dispatches the closest available unit to each call for service (implying use of a perfect resolution AVL system). The model accommodates a realistic description of the service area and rather general spatial deployment policies for units.

In implementing the model for efficient computer execution, the focus is on computation and storage minimizing procedures for generating the state-to-state Markov transition rates. One useful technique involves the effective application of a recently developed backward regenerative unit-step tour of the hypercube. The algorithmic procedures generalize to computer solutions of $M|M|N$ queuing systems with distinguishable servers, different customer classes, and a cost structure for assigning servers (who may be in one of several "postures") to customers of each class.

The paper concludes with a realistic nine-unit police example that indicates the general ways in which AVL dispatching improves (and degrades) system performance.

1. INTRODUCTION AND OVERVIEW

In urban emergency service systems (e.g., police, fire, ambulance and emergency repair services), increased attention is being focused on various technological innovations for improving operational performance. Automatic vehicle location (AVL) systems comprise one important class of such innovations. These systems would present to the radio dispatcher the estimated locations of all emergency response units under his jurisdiction. Dispatch decisions could then be made with an awareness of these estimated locations, resulting in improved operations compared to standard manual dispatching procedures.

This paper presents a method for modeling analytically one simple form of dispatching using AVL information. It is assumed that the *exact*, realtime locations of all response units are known (i.e., a perfect resolution AVL system) and that the dispatcher always dispatches the *closest available* response unit to an incident (closest vehicle dispatching).

*Richard C. Larson is Associate Professor of Urban Studies and Electrical Engineering, Massachusetts Institute of Technology. He holds a B.S., M.S. and Ph.D. in electrical engineering, with specialization in operations research, from M.I.T. Professor Larson's papers have appeared in *The Journal of Computers and Operations Research, Operations Research, Management Science, Journal of Criminal Justice, IEEE Transactions on Systems Science and Cybernetics, IEEE Transactions on Vehicular Technology, Journal of Urban Analysis, Sloan Management Review, Evaluation,* and *Journal of Research on Crime and Delinquency.* He is author of a book *Urban Police Patrol Analysis*, M.I.T. Press, 1972, which was awarded the 1972 Lanchester Prize of ORSA. He has served as a member of the Science and Technology Task Force of the President's Commission on Law Enforcement and Administration of Criminal Justice (1966–67) and the Police Advisory Panel of the National Commission on Productivity (1973).

†Evelyn A. Frank, now a consultant in Paris, France, received her Masters Degree in Operations Research from the Massachusetts Institute of Technology in 1976. She received the S.B. degree from Universidad de los Andes (Bogota, Colombia) in 1973. At the Universidad de los Andes she held the position of lecturer on Computer Programming and Research Fellow in Planning.

The model builds on "the hypercube queuing model," which is a spatially distributed queuing model developed recently to analyze analytically the performance of urban emergency services[1]. Various algorithms and theoretical concepts of the hypercube model are extended to incorporate in an efficient manner the more complicated dispatching mechanism presented by the AVL system.

The hypercube implementation of the closest available vehicle strategy computes all of the standard hypercube performance measures: workloads of each of the units, mean travel times and cross-area dispatch frequencies. In addition, it computes new measures that relate specifically to AVL systems: point-specific dispatch error probabilities for any non-AVL (fixed preference) strategy and point-specific mean travel time reductions due to AVL.

AVL systems are just now being implemented in U.S. cities. The St. Louis (Missouri) Metropolitan Police Department has allocated $2.7 million toward implementing a computer-assisted dead reckoning system, utilizing funding from the Law Enforcement Assistance Administration of the U.S. Department of Justice. [12] [14] This system became fully operational (city wide) in March 1977. Also in 1977 the City of Huntington Beach, California, implemented a fixed-post sensor system for use by its police department and (eventually) its other emergency services. Other police departments utilizing an AVL-system include those of Montclair, California; Stamford, Connecticut; and Dallas, Texas. Numerous other cities have expressed interest in AVL systems. Additional AVL test and implementation work is being supported by the U.S. Department of Transportation, using sites in Philadelphia, Pennsylvania and Los Angeles, California.

Given the current attention directed toward AVL systems, it is especially important for potential users to have access to an inexpensive tool for evaluating the consequences of AVL in their own cities. For emergency services the hypercube model (with AVL dispatching) is intended to fulfill this need. It allows detailed examination of the projected "before and after" effects of closest unit dispatching, thereby facilitating an evaluation of AVL prior to purchase and implementation. This is especially critical due to the relatively large purchase price (and associated upkeep cost) of AVL systems, often amounting to millions of dollars per city. It is also important that the model includes details of a city's geography, travel time characteristics, and spatial deployment patterns; the operational effects of AVL dispatching depend in a complicated way on each of these factors. The *PL/I* computer program which implements the model described herein is available (at cost) from the address contained in [13] (which is the user's manual for the program).

Prior to 1972, two AVL modeling efforts are noteworthy. The first was performed by M. Bellmore as part of the work of the Science and Technology Task Force of the President's Commission on Law Enforcement and Administration of Justice[2]. Bellmore specified a probability ρ that any particular unit would be unavailable to respond to a call for service; higher levels for ρ indicate a patrol force with higher workloads. Units were determined to be available or busy by independent Bernoulli trials, with ρ being the busy probability. A square simulated region contained N regularly spaced square patrol beats, the length of each side being $\sqrt{N}$ (assumed integer) beat lengths. Position estimation resolution of the AVL system was specified by a fraction $1/r$ (r integer), such that for a given r each beat was partitioned into r^2 square subbeats. The AVL system would specify the subbeat of the patrolling unit with certainty, but the location of the unit within the subbeat was assumed to be uniformly distributed. The dispatcher using the AVL system would always dispatch the vehicle estimated to be closest according to the right angle distance metric. This metric specifies that the distance between two points (x_1, y_1) and (x_2, y_2), assuming directions of travel are parallel to the coordinate axes, is $d_{12} = |x_1 - x_2| + |y_1 - y_2|$. In using the right angle distance metric, the unit's position was "guesstimated" by the dispatcher (actually a computer subroutine) to be at the center of its subbeat. Bellmore's analysis focused on travel time savings achievable with AVL as a function of workload ρ and resolution r. While a pioneering effort in the analysis of AVL systems, Bellmore's model was limited in several areas: (1) the actual way that units become busy is not according to independent Bernoulli trials, and there is significant dependence among states of response units; (2) it is not apparent how to apply Bellmore's ideas to realistic nonsymmetric situations, perhaps involving overlapping beats; (3) being a simulation model, it was impractical with Bellmore's model to compute point-specific (as well as area-

averaged) performance measures; (4) the resolution model, in terms of subbeats, does not reflect the workings of any actual AVL system.

The second effort by Bales[3] used most of Bellmore's ideas but with a resolution model that was found realistic for most radio-trilateration AVL systems. This modeled the position estimation error as a circularly symmetric Gaussian error, where the peak of the Gaussian distribution was located at the response unit's location.

In 1972 Larson devoted a chapter of his book to AVL systems ("Evaluating Technological Innovations: Automatic Car Locator Systems" in [4]). The analysis included "back-of-the-envelope" geometrical probability models of simple one- and two-dimensional situations and standard queuing models operating with AVL dispatching. It also included a simulation model, incorporating a circulatory symmetric Gaussian error characteristic, to depict realistic urban situations.

Recent technological advances in AVL development have revealed systems whose resolution is well within a fifth or less of a beat length. Bellmore's and Bale's work, as well as Larson's work in 1972, indicated that systems with such good resolutions achieve response time reductions almost as great as perfect resolution systems. Thus, for modeling the operational impacts upon an emergency service system (most likely a police department) it is not unreasonable to assume a perfect resolution system.

This is the approach taken with the hypercube model, namely a perfect resolution AVL system is assumed. The limitations of the earlier modeling work regarding the lack of independence of the states of response units, the adaptability to arbitrary (nonsymmetric, realistic) situations, and the computation of point-specific performance measures are overcome within the hypercube framework. Given this framework, one models the geography of a region as a set of discrete points called reporting areas or geographical atoms. A mobile response unit, typically a police car, can be located in any one of a subset of these points, called a district, with an arbitrary probability distribution over the atoms in the district. In police applications a district may in fact be called a beat or sector or route or area. Districts may overlap in arbitrary ways, reflecting rather complicated spatial deployment policies. An example with $N = 5$ response units with nonoverlapping districts is given in Fig. 1. Here a call for service arrives from an atom in the northeast quadrant of district 3, finding all five response units available at

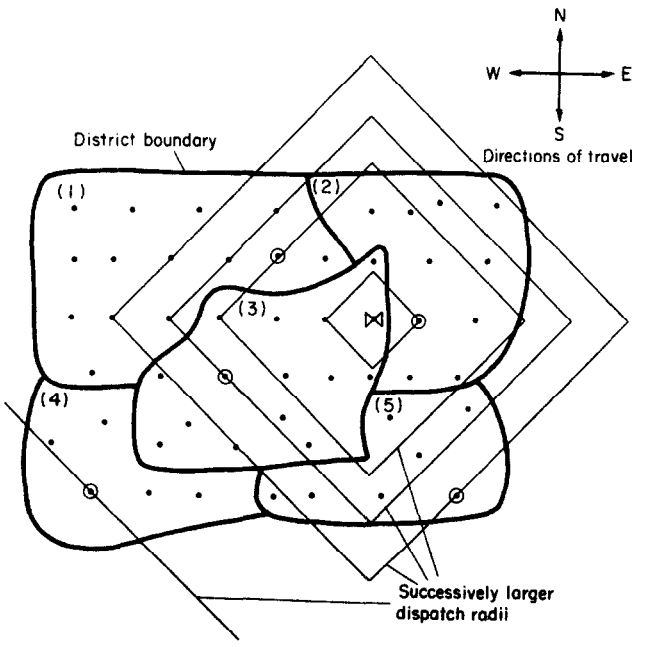

Fig. 1. $N = 5$ Example in which the district unit is not the closest available unit.

the locations (atoms) indicated. A dispatcher operating under a non AVL dispatching strategy, would most likely dispatch unit 3 to the scene of the call for service. However, assuming right angle travel distances, unit 2 is found to be closest to the scene, followed by unit 1 and then unit 3. Thus, with this configuration of units and this particular atom for the call for service, the AVL strategy implemented within the hypercube model would dispatch unit 2, yielding a substantial savings in travel time or distance.

A non AVL strategy is usually described by a fixed vector of preferred units for each atom generating a call for service, and the dispatcher assigns the *first available* unit in the vector (starting with the most preferred unit, which most likely would have been the district 3 unit in this example). An AVL strategy is a matrix strategy, with an element in the matrix being the conditional probability that some unit n will be the ith preferred unit, given that all units are available for dispatching; if certain units are not available, then the conditional probabilities must be recomputed and scaled accordingly. Given a specified availability of units and a location for the call, if the non AVL strategy yields a dispatch decision different from the AVL strategy (that is, if it selects other than the closest available unit), then a *dispatch error* is said to occur. The hypercube model computes numerous performance measures for the system, including region-wide and point-specific dispatch error probabilities. This allows detailed analysis of any fixed preference (non AVL) strategy vs an AVL strategy.

After briefly reviewing the relevant terminology of the hypercube model, the paper focuses on the efficient computation of state- and unit-dependent rates of transition to other system states. Exploiting and expanding certain ideas applied to earlier implementations of the hypercube model, an efficient procedure is found for computing the upward transition rates (i.e., those that determine frequencies of dispatch of each vehicle) without ever storing the usually huge dispatch probability matrix. Part of the procedure involves a unit-step tour of the hypercube vertices. The methods are illustrated with a simple computational example.

Once the state-to-state transition rates are found, the hypercube model is "solved" in the usual way[1], where solution implies numerical computation of the steady state system probabilities. Then these probabilities are coupled with a second hypercube tour, detailed in Section 5, to compute system performance measures. A nine-unit example, first studied with a simulation model[4], is then analyzed in Section 6. Certain point-specific performance measures for the nine-unit example are available for the first time. A final discussion section indicates how the algorithmic procedures used here apply to a rather general class of queuing system.

For convenience, a summary of frequently used symbols is given in Table 1. Those desiring additional background on AVM systems may wish to consult AVL [5] and [6].

2. BRIEF DESCRIPTION OF THE HYPERCUBE MODEL

Before describing the method used to model AVL dispatching, we give a brief overview of the relevant hypercube terminology. A more detailed description is given in Ref. [1].

The hypercube model assumes a geographical region R which is quantized into J geographical areas or *atoms*. Associated with each atom j $(1 \le j \le J)$ is the fraction of region-wide workload f_j generated from within the atom $\left(\sum_{j=1}^{J} f_j = 1\right)$. Central to dispatching decisions, the variable τ_{ij} is the mean travel time from atom i to atom j.

For the region R, there is a set of N response units to respond to calls for service from the atoms. While available, these response units can be mobile and their possible locations are given by the location matrix $L = (l_{nj})$, where l_{nj} is the probability that response unit n is located in atom j while available $\left(\sum_{j=1}^{J} l_{nj} = 1 \text{ for all } n\right)$. The set of possible locations per unit n (i.e., those j for which $l_{nj} > 0$) is called the unit's *district*; note that in general, districts may overlap.

From a queuing point of view we assume that customers (calls for service) are generated from within the region in a Poisson manner at a mean rate λ per hr., with each atom j acting as an independent Poisson generator with mean rate λf_j. Given the arrival of a call for service, exactly one of the servers (response units) is dispatched to its location, assuming at least one server is available. For the case of a zero-line capacity queue, any call for service that arrives while all N response units are busy is either lost or serviced from outside the region or by special reserve units from within the region. For the case of an infinite line capacity queue, any

Table 1. Summary of frequently used symbols

N	Total number of response units
J	Total number of geographical atoms
f_j	Fraction of region-wide workload generated from atom j, $j = 1, 2, \ldots, J$
τ_{ij}	Mean travel time from atom i to atom j, $i,j = 1, 2, \ldots, J$
l_{nj}	Probability that response unit n is located in atom j while available, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$
λ	Average number of calls for service generated per hour from within the entire region of interest
μ^{-1}	Average service time per call
C_N	Vertices of N-dimensional unit hypercube in the positive orthant
B	A vertex contained in C_N
b_n	The nth binary digit (from the right) in B, $n = 1, 2, \ldots, N$
$w(B)$	Weight of vertex $B(\Sigma b_n)$
$v(B)$	Numerical value of vertex B
d_{ij}	Hamming distance between two vertices B_i and B_j
λ_{ij}	Infinitesimal mean rate at which transitions are made from state i to state j, given the system is in state i ($i \neq j$)
Λ	Matrix of infinitesimal transition rates (with $\lambda_{ii} \equiv -\sum_{j \neq i} \lambda_{ij}$)
S_k	kth member of unit-step hypercube tour, $k = 1, 2, \ldots, 2^N$
t_{nj}	Mean time for unit n to travel to atom j, given unit n is available, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$
ρ_n	Workload of unit n (measured in fraction of time busy servicing calls), $n = 1, 2, \ldots, N$
ρ_{nj}	Fraction of all dispatches that send unit n to atom j, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$
$P_{n\|jk}$	Prob{unit n is the closest available unit|call in atom j, state of the system is B_k}, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$; $k = 0, 1, \ldots, 2^N - 1$
$q_{nr\|jk}$	Prob{unit n is the closest available unit *and* is located in atom r|call in atom j, state of the system is B_k}, $n = 1, 2, \ldots, N$; $r,j = 1, 2, \ldots, J$; $k = 0, 1, \ldots, 2^N - 1$
$t_{n\|jk}$	Expected travel time of unit n, given that call in atom j, state of the system is B_k, and unit n is dispatched, $n = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$; $k = 0, 1, \ldots, 2^N - 1$
$r(e)$	Atom number of eth closest atom to some specified atom, $e = 1, 2, \ldots, J$
$\mathscr{A}_k$	Set of available units, given the state of the system is B_k, $k = 0, 1, \ldots, 2^N - 1$

call for service that arrives while all N response units are busy is entered at the end of a queue of calls that is depleted by any simple queuing discipline that ignores the incident's location (e.g., first come, first served, random). The service time of any response unit for any call for service is assumed to have a negative exponential distribution with mean $1/\mu$. (It is straightforward to generalize the results to allow each server n to have its own mean service time $1/\mu_n$. Moreover, recent computational experience with other service time distributions indicates a significant insensitivity of results of the model to the exact form of the service time distribution[7].) Since the service time is assumed to be independent of the identity of the server, the location of the customer, and the history of the system, variations in the service time that are due solely to variations in travel times are ignored (mean travel time being assumed to be an order of magnitude smaller than mean on-scene service time).

A state of the system is denoted by an ordered set of N binary digits $B = \{b_N, b_{N-1}, \ldots, b_1\}$, where unit n is said to be *busy* if $b_n = 1$ and *idle* (or *available*) if $b_n = 0$. The weight of B, denoted $w(B)$, is equal to $\Sigma\, b_n$, the number of binary "ones" in the set B. The numerical value associated with the set B is

$$v(B) = \sum_{n=1}^{N} b_n 2^{n-1}.$$

To each set $B = \{b_N, b_{N-1}, \ldots, b_1\}$ there corresponds a unique point or *vertex* (or *state*) in R^N with nth coordinate equal to b_n ($n = 1, \ldots, N$). The set C_N of all 2^N such vertices is the set of vertices of the N-dimensional unit hypercube in the positive orthant.

Defining binary set operations in the usual way, the *Hamming distance* between two vertices B_i and B_j is the weight of the symmetric set difference

$$d_{ij} \equiv w([B_i \cap B_j'] \cup [B_i' \cap B_j]).$$

For example, two vertices that are unit Hamming distance apart differ in only one binary digit. The "upward" Hamming distance is defined to be

$$d_{ij}^{+} \equiv w(B_i' \cap B_j)$$

and the "downward" distance

$$d_{ij}^{-} \equiv w(B_i \cap B_j'),$$

where

$$d_{ij} = d_{ij}^{+} + d_{ij}^{-}.$$

The state space of the zero-line capacity model is C_N, where each vertex (state) corresponds to a particular combination of response units busy and idle. This state space is augmented by an "infinite tail" for the infinite-line capacity case. Since the two models are governed by the same equations for unsaturated states (i.e., states with at least one available response unit), we will focus the development on just one of the models: the zero-line capacity model. The infinite-line capacity case requires modifications for dispatches from a queue identical to those described in [1].

The state transition matrix is $\Lambda = (\lambda_{ij})$, where $\lambda_{ij} =$ infinitesimal mean rate at which transitions are made from state i to state j, given that the system is in state i; $i,j = 0, 1, \ldots, 2^N - 1$, $i \neq j$, and $\lambda_{ii} = -\sum_{j \neq i} \lambda_{ij}$. Here for convenience we index the states according to their numerical values, i.e., we select i so that $v(B_i) = i$, $i = 0, 1, \ldots, 2^N - 1$.

There are two classes of transitions on the hypercube: *upward* transitions that change a unit's status from available to unavailable and *downward* transitions that do the reverse. For a given vertex $B_i = \{b_N, b_{N-1}, \ldots, b_1\}$, upward transitions can occur to all "adjacent" vertices B_j for which $d_{ij}^{+} = 1$. If unit n is the unit whose status is changed (from idle to busy), then $B_i = \{b_N, b_{N-1}, \ldots, 0_n, \ldots, b_1\}$ and $B_j = \{b_N, b_{N-1}, \ldots, 1_n, \ldots, b_1\}$. Downward transitions can occur to all adjacent vertices B_j for which $d_{ij}^{-} = 1$. No transitions occur to vertices that are more than unit Hamming distance from B_i since only one unit is assigned to each call.

Since the service times are all distributed as negative exponential random variables with mean μ^{-1}, the transition rate associated with each downward transition is equal to μ. Thus, for all (i, j) for which $d_{ij}^{-} = 1$ we have $\lambda_{ij} = \mu$. For convenience we set $\mu = 1$, thereby equating the unit of time to the mean service time.

The upward transition rates depend in a complicated way on the region's geography, the system state and (for AVL dispatching) the real-time locations of the available units. Following the earlier work with fixed preference dispatching strategies, we will develop a recursive method to generate the set of upward transition rates, first by fixing the geographical atom of the call, then by *touring* the hypercube in a *unit-step* fashion. The entire matrix is completed as soon as the hypercube has been toured once for every geographical atom.

As argued in [1], the model is a *finite-state continuous time Markov process* whose steady-state probabilities are determined from the *equations of detailed balance*,

$$P\{B_j\}[\lambda_j + w(B_j)] = \sum_{\{B_i \in C_N : d_{ij}^{+} = 1\}} P\{B_i\}\lambda_{ij} + \sum_{\{B_i \in C_N : d_{ij}^{-} = 1\}} P\{B_i\}, \quad j = 0, 1, \ldots, 2^N - 1 \tag{1}$$

where $P\{B_j\} \equiv$ Prob{system is occupying state j under steady-state conditions},

$$B_j \in C_N,\ j = 0, 1, \ldots, 2^N - 1; \quad \text{and} \quad \lambda_j = \begin{cases} 0 \text{ for } j = 2^N - 1, \\ \lambda \end{cases}$$

To guarantee a probability distribution, we also require that the probabilities sum to one, i.e.,

$$\sum_{i=0}^{2^N - 1} P\{B_i\} = 1, \tag{2}$$

a condition which makes any one of the balance equations redundant and therefore removable from the set of equations.

3. COMPUTING THE UPWARD TRANSITION RATES FOR AN AVL STRATEGY

As in the hypercube model with fixed preference dispatching, it is necessary to develop a time- and storage-efficient way of generating the upward transition rates. For each atom j

$(1 \le j \le J)$ we shall tour the hypercube in a unit-step fashion, visiting in succession the states $S_1, S_2, \ldots, S_{2^N}$. From one step to the next, the status of only one response unit changes, either from busy to idle or idle to busy. The procedure for generating the upward transition rates avoids the requirement of storing usually huge intermediate matrices.

Unlike the model with fixed preference policies, the AVL model requires a particular unit-step tour to be traversed. This is the *backward regenerative unit-step tour* $S_1, S_2, \ldots, S_{2^N}$ derived in [1]. Thus any particular hypercube vertex has an associated B_i such that $i = v(B_i)$, $i = 0, 1, 2, \ldots, 2^N - 1$, and an S_k such that the vertex is the kth visited vertex in the unit-step backward regenerative unit-step tour, $k = 1, 2, \ldots, 2^N$. Note that $S_k = B_{v(S_k)}$. To generate this tour for N units, one supposes that the tour for $N-1$ units is known, the tour for one unit being {0}, {1}. The first 2^{N-1} entries in the larger tour are obtained simply by augmenting the entries in the smaller tour with a zero ("0") in the Nth digit (from the right). The final 2^{N-1} entries are obtained by augmenting the entries in the smaller tour with a one ("1") in the Nth digit *and* reversing the order of the sequence. As an example, after carrying out this procedure twice, starting with the one-unit tour, we obtain the 3-digit tour: $S_1 = \{0, 0, 0\} = B_0$, $S_2 = \{0, 0, 1\} = B_1$, $S_3 = \{0, 1, 1\} = B_3$, $S_4 = \{0, 1, 0\} = B_2$, $S_5 = \{1, 1, 0\} = B_6$, $S_6 = \{1, 1, 1\} = B_7$, $S_7 = \{1, 0, 1\} = B_5$, $S_8 = \{1, 0, 0\} = B_4$.

We now focus on computing the upward transition rates. Note that in general any unit n can be dispatched to a call in atom j, provided it can possibly be closer than any other available unit(s). Such multiple-choice assignments from the same state to the same atom occur with fixed preference policies only in the case of ties in dispatch preferences. In order to compute incrementally each upward transition rate, we must narrow our focus to the particular atom containing the dispatched unit, thereby defining

$$q_{na|jk} \equiv \text{Prob}\{\text{unit } n \text{ is the closest available unit and is located in atom } a | \text{call is in atom } j \text{ and the state of the system is } B_k\}.$$

For all states in which there is at least one available unit, we must have a proper probability distribution:

$$\sum_{n=1}^{N} \sum_{a=1}^{J} q_{na|jk} = 1 \qquad j = 1, \ldots, J; \quad k = 0, 1, \ldots, 2^N - 2.$$

For $k = 2^N - 1$ we have $B_k = \{1, 1, \ldots, 1\}$, implying that all units are busy; in that case calls are either lost or handled by a back-up service, and

$$q_{na|j(2^N-1)} = 0 \qquad \text{for all } n, a, j.$$

Since the hypercube upward transition rates are aggregated over possible unit locations, we must also define

$$P_{n|jk} \equiv \text{Prob}\{\text{unit } n \text{ is the closest available unit} | \text{call is in atom } j \text{ and the state of the system is } B_k\}.$$

Clearly,

$$P_{n|jk} = \sum_{a=1}^{J} q_{na|jk}. \tag{3}$$

Now, the upward transition rates λ_{ik} are calculated as follows: The matrix Λ is initialized to zero. Then when the point of the tour is reached where the call is in atom j and the state of the system is $B_i = \{b_N, b_{N-1}, \ldots, b_1\}$, the transition rates to *all* the adjacent states $B_k = \{b_N, b_{N-1}, \ldots, b'_n, \ldots, b_1\}$ such that $d_{ik}^{+} = 1$ and $b'_n = 1 \neq b_n$ will be incremented by $\lambda f_j P_{n|ji}$. The quantity $\lambda f_j P_{n|ji}$ is the component of the total upward transition rate out of state B_i corresponding to calls from atom j that result in unit n becoming busy (transitioning to state B_k). The incrementing is

given by

$$\lambda_{ik} \leftarrow \lambda_{ik} + \lambda f_j P_{n|ji}. \tag{4}$$

In many cases $P_{n|ji} = 0$ for some n and no operation is performed.

We start the tour in state $S_1 = B_0 = \{0, 0, \ldots, 0\}$ implying that all N units are available for dispatch. Our initial focus is toward computing $q_{na|j0}$, then toward updating the computation as unit-step transitions are made from state to state.

To generate the $q_{na|jk}$ efficiently, we rank all atoms i for each unit n for which $l_{ni} > 0$, in order of increasing travel time to atom j. After doing this for all n, we merge the rankings into a global ranking, defining the eth entry in the global ranking to be the eth closest possible unit location to atom j. In the case of two or more units possibly occupying the same location or atom (implying overlapping districts) or in the case of two or more different atoms being the same travel time to atom j, the e-ordering is assigned arbitrarily and any resulting tie(s) will be broken by random choice. For the eth entry in the ranking, atom $r(e)$ is the corresponding unit location and $l_{nr(e)}$ is the probability that unit n is in atom $r(e)$ while available.

The minimum ranking for unit n is

$$L_{nj} = \min_{\{\text{all } e : l_{nr(e)} > 0\}} e.$$

The maximum ranking is

$$U_{nj} \equiv \max_{\{\text{all } e : l_{nr(e)} > 0\}} e.$$

Thus, possible locations for unit n can range from the L_{nj}'th entry in the table to the U_{nj}'th entry. For the case of collectively exhaustive nonoverlapping districts, this range starts with the L_{nj}'th closest atom to atom j and ends with the U_{nj}'th closest atom to atom j. The L_{nj}'s will provide a guide for ordering the response units for use in the backward regenerative unit-step tour of the hypercube vertices.

Ignoring ties (for the moment), we compute $q_{na|jk}$ by arguing as follows:

$$q_{na|jk} = \text{Prob}\{\text{no available unit is closer than } \tau_{aj} \text{ to atom } j\} \cdot \text{Prob}\{\text{unit } n \text{ is in atom } a | \text{unit } n \text{ is not in any closer atom}\}.$$

Setting $a = r(e)$, the first term on the right hand side is simply

$$P_{\geq e} = 1 - \sum_{m=1}^{N} \sum_{f=1}^{e-1} q_{mr(f)|jk}, \tag{5}$$

and the second term is

$$l^c_{nr(e)} = l_{nr(e)} \Big/ \left(1 - \sum_{f=1}^{e-1} l_{nr(f)}\right), \tag{6}$$

the denominator arising from the conditional information that unit n is not closer than atom $r(e)$. Thus, the equation that is useful in iterative computations is

$$q_{nr(e)|jk} = P_{\geq e} l^c_{nr(e)} = \left[1 - \sum_{m=1}^{N} \sum_{f=1}^{e-1} q_{mr(f)|jk}\right] \left[l_{nr(e)} \Big/ \left(1 - \sum_{f=1}^{e-1} l_{nr(f)}\right)\right]. \tag{7}$$

Note that $l^c_{nr(e)}$ need only be computed once for any atom j and $P_{\geq e}$ is found simply by subtracting the total accumulated probability (through table entry $e-1$) from one.

Compaction

Suppose, given full availability of units, that some unit n may be located in two or more *consecutive* atoms $r(e)$, $r(e+1)$, $r(e+2)$, ..., $r(e+m)$. Then if only *average* travel time (rather than its distribution) is to be computed with the model, the $m+1$ entries in the e-ordered table can be replaced by one aggregate entry. Call the new "aggregate atom" $r(e \rightarrow m)$. Then the new

entry in the table is specified by

$$l_{nr(e\to m)} = \sum_{i=e}^{e+m} l_{nr(i)} \qquad \tau_{r(e\to m)j} = \sum_{i=e}^{e+m} \tau_{r(i)j} l_{nr(i)} / l_{nr(e\to m)}.$$

This compacted entry remains applicable to all 2^N system states, often yielding substantial savings in computation time.

Ties

Suppose in the process of computing $q_{nr(e)|jk}$ iteratively η ($\eta \geq 2$) units are found which could be tied for dispatch preference. In other words, up to η units could be equidistant from atom j, located either in the same atom i or in different atoms equidistant from atom j. In such a case the e ranking for the tied units can be assigned arbitrarily. Then, if one wishes to treat ties inexactly, they could be ignored by applying equation (7) directly, implying that a tie would be broken in the same (arbitrary) way each time that it is incurred.

If an exact treatment is desired, let $e_{\eta-}$ denote the ranking of the last untied unit location. Let the tied units be denoted by $s_1, s_2, \ldots, s_\eta$, corresponding to unit locations ranked $e_{\eta-}+1$, $e_{\eta-}+2, \ldots, e_{\eta-}+\eta$. Then for some unit s_i equation (7) is replaced with

$$q_{s_i r(e_{\eta-}+i)|jk} = P_{\geq e_{\eta-}} Q(e_{\eta-}, s_i),$$

where

$Q(e_{\eta-}, s_i) \equiv$ Prob{unit s_i is dispatched|a possible tie among units $s_1, s_2, \ldots, s_\eta$ and no unit dispatched in any unit location ranked $e_{\eta-}$ or lower}.

Allowing all possible combinations, we have

$$Q(e_{\eta-}, s_i) = l^c_{s_i r(e_{\eta-}+i)} \prod_{m\neq i} (1 - l^c_{s_m r(e_{\eta-}+m)}) + \frac{1}{2} l^c_{s_i r(e_{\eta-}+i)} \sum_{\substack{k=1 \\ k\neq i}}^{\eta} l^c_{s_k r(e_{\eta-}+k)} \prod_{m\neq i,k} (1 - l^c_{s_m r(e_{\eta-}+m)}) + \cdots + \frac{1}{\eta} \prod_{m=1}^{\eta} l^c_{s_m r(e_{\eta-}+m)}.$$

While this equation is tedious to compute for general η, in practice one rarely incurs ties of order higher than two or three.

Transitioning to adjacent states

Suppose at some unit-step tour state $S_{k'-1}$ we have computed $q_{na|jv(S_{k'-1})}$ for all n and a. For instance, at state S_0 we would have computed $q_{na|j0}$. Then we make a unit-step transition to an adjacent tour state $S_{k'} = B_{v(S_{k'})}$, causing some unit m to change status. For notional simplicity let $k = v(S_{k'})$. Now for state B_k we want to compute $q_{na|jk}$ for all n and a, taking advantage of our previous computations wherever possible.

If the smallest possible e ranking for unit m ($l_{mr(e)} > 0$) is such that there is always an available unit n closer to atom j than unit m, then the change in status of unit m does not affect the other units and all the $q_{na|jk}$ remain unchanged from the previous tour state. On the other hand, if unit m could possibly be the closest to atom j, then some computations must be performed. Thus, letting $\mathscr{A}_k$ = set of available units in state B_k, there are two cases:

1. $L_{mj} > \min_{\substack{\{n\in\mathscr{A}_k\} \\ n\neq m}} U_{nj}$

In this case all $q_{na|jk}$ remain unchanged.

2. $L_{mj} \leq \min_{\substack{\{n\in\mathscr{A}_k\} \\ n\neq m}} U_{nj}$

In this case all $q_{nr(e)|kj}$ such that $e < L_{mj}$ remain unchanged (since changing the status of unit m

has no effect for units located in atoms closer than the unit can be located). All $q_{nr(e)|jk}$ such that $e \geq L_{mj}$ are recomputed applying equation (7).

Reordering the response units

In traversing the hypercube in a unit-step fashion, the amount of computation to be performed will depend in large part on the frequency with which we encounter case 1: $L_{mj} > \min_{n \text{ available}} U_{nj}$, implying all the $q_{na|jk}$ are unchanged from the last toured vertex. The more often we encounter this case, the less will be the computational effort. Examining the backward regenerative unit-step tour, and recalling that the state of unit n is specified by the nth binary digit from the right, we see that in a complete tour unit n changes status 2^{N-n} times. Thus, by making binary digit n correspond to that unit m for which L_{mj} is the nth largest of the L_{mj}'s, we change the status most frequently of units distant from atom j (resulting in case 1 occuring much more frequently than would occur with an arbitrary ordering of response units). Thus, to summarize, we reorder the response units as follows:

Unit 1 is that unit with largest L_{mj}.
Unit 2 is that unit with second largest L_{mj}.
.
.
.
Unit N is that unit with smallest L_{mj}.

We now briefly analyze the efficiency of the combined procedure of performing the unit-step tour and reordering the response units when computing $q_{na|jk}$ for a fixed j. Suppose, in the worst case, that one simply recomputed $q_{na|jk}$ exhaustively for each different system state B_k. The number of different values for $q_{na|jk}$ for fixed j and k is $\underset{m}{\text{MAX}}\, U_{mj} - \underset{m}{\text{MIN}}\, L_{mj} + 1$. Since these values would have to be recomputed for each state, the total number of computations of $q_{na|jk}$ required per complete set of states would be bounded above by

$$[\underset{m}{\text{MAX}}\, U_{mj} - \underset{m}{\text{MIN}}\, L_{mj} + 1]2^N,$$

since there are 2^N different states. If one does not count "zeroes" as computations, due to units that are busy (and thus cannot be dispatched), then a better approximation for the number of times $q_{na|jk}$ must be computed is

$$[\underset{m}{\text{MAX}}\, U_{mj} - \underset{m}{\text{MIN}}\, L_{mj} + 1]2^{N-1},$$

since each unit is available (nonbusy) at half of the vertices. If one also wishes to discount "zeroes" due to exceeding the smallest U_{mj} of the currently available units for a given state B_k, then perhaps the best approximation of the number of computations of $q_{na|jk}$ required per complete set of states is given by

$$NC_1 \approx \frac{1}{N} \sum_{n=1}^{N} (U_{nj} - L_{nj} + 1)2^{N-1}, \tag{8}$$

where this average range is used to reflect the fact that as soon as the range is exceeded for an available unit, all remaining $q_{na|jk}$ are zero.

We now quote a formula from the Appendix for the number of computations of $q_{na|jk}$ required using the backward regenerative unit step tour in combination with reordering the units according to L_{mj}. To avoid excessive complexity, we allow no ties and require that the upper bounds be ranked in the same way as the lower bounds. So, given the reordering of units, we must have

$$L_{Nj} < L_{(N-1)j} < \cdots < L_{1j}$$
$$U_{Nj} < U_{(N-1)j} < \cdots < U_{1j}.$$

Define the positive part of X as follows:

$$(X)_+ = \begin{cases} X & \text{if } X > 0 \\ 0 & \text{if } X \le 0 \end{cases}$$

Then the number of computations of $q_{na|jk}$ required (per tour) using the backward regenerative unit step tour with reordered units is

$$NC_2 = \sum_{m=1}^{N} \sum_{n=m+1}^{N} (U_{nj} - L_{mj} + 1)_+ 2^{n-m-1} + \sum_{m=1}^{N} (U_{mj} - L_{mj} + 1). \tag{9}$$

In applying (9) in practical situations (e.g., data describing the district of a police department of one large city) one usually experiences reductions in computational effort (compared to the average range formula given by equation 8) of at least 4-to-1 and sometimes up to 10-to-1.

As a further remark on efficiency, it is noteworthy that the matrix dispatching strategy implied by AVL dispatching requires no more storage of dispatching information than is required for fixed preference dispatching; and this is $(N+1)2^N$ nonzero matrix elements, as argued in [1]. The process of incrementing the (aggregated) upward transition rates during the hypercube tour, as the $q_{na|jk}$ are computed, implies that neither $q_{na|jk}$ nor $P_{n|jk}$ need be stored after leaving state B_k. This is indeed fortunate since $q_{na|jk}$ could require an array of $NJ^2 2^N$ elements to store completely; for $N = 10$ units and $J = 100$ atoms, this is about 10^8 storage elements, far too many to make the procedure practical.

4. ILLUSTRATIVE COMPUTATIONAL EXAMPLE

To illustrate the procedures described in Section 3, we consider the simple $N = 3$ unit, $J = 16$ atom example summarized in Table 2. Here the atom of the incident is fixed at $j = 1$. We wish to compute $q_{nr(e)|1k}$ as specified in equation (7). From Table 2, note that units have been numbered according to lower bounds L_{n1}, with $L_{31} = 1$, $L_{21} = 5$, and $L_{11} = 11$; atoms have been ordered by the index e according to increasing travel times $\tau_{r(e)1}$.

The procedure for computing $q_{nr(e)|1k}$ starts in state $B_0 = \{0, 0, 0\}$. Applying (7) for $e = 1, 2, 3, 4$, it is apparent, due to no overlap with unit 2, that $q_{3r(e)|10} = l_{3r(e)}$. Subtracting the sum of the probabilities from one at that point, we have $P_{\ge 5} = 1 - (0.25 + 0.133 + 0.167 + 0.200) = 0.25$. Thus $q_{2r(5)|10} = 0.25(0.364) = 0.091$. Now $P_{\ge 6} = 0.159$, and thus $q_{3r(6)|10} = 0.159(0.133/(1 - [0.75])) \approx 0.085$. Continuing this process yields $q_{1r(7)|10} \approx 0.032$ and $q_{1r(8)|10} \approx 0.042$. But $e = 8 = U_{11}$, and thus computations are completed for state B_0.

Table 2. Illustrative hypercube tour: $N = 3$ units, $J = 16$ atoms, incident in atom $j = 1$

e	atom $r(e)$	$\tau_{r(e)1}$	Unit n	$l_{nr(e)}$	B_0 000 $q_{nr(e)\|10}$	B_1 001 $q_{nr(e)\|11}$	B_3 011 $q_{nr(e)\|13}$	B_2 010 $q_{nr(e)\|12}$	B_6 110 $q_{nr(e)\|16}$	B_7 111	B_5 101 $q_{nr(e)\|15}$	B_4 100 $q_{nr(e)\|14}$
$1 = L_{31}$	1	0.0	3	0.25	0.25	0.25	0.25	0.25 ●				
2	6	3.7	3	0.133	0.133	0.133	0.133	0.133				
3	5	4.8	3	0.167	0.167	0.167	0.167	0.167				
4	2	5.9	3	0.20	0.200	0.200	0.200	0.200				
$5 = L_{21}$	7	8.5	2	0.364	0.091	0.091 ●	0.0	0.0			● 0.364	0.364
6	4	8.7	3	0.133	0.085	0.085	0.133	0.133				
7	8	9.2	2	0.273	0.032	0.032	0.0	0.0			0.273	0.273
$8 = U_{31}$	3	11.5	3	0.117	0.042	0.042	0.117	0.117				
9	10	13.2	2	0.227							0.227	0.227
$10 = U_{21}$	9	13.9	2	0.136							0.136	0.136
$11 = L_{11}$	11	17.2	1	0.130		●		●	0.130 ●			●
12	12	17.7	1	0.098					0.098			
13	13	20.2	1	0.043					0.043			
14	14	20.4	1	0.087					0.087			
15	15	22.8	1	0.217					0.217			
$16 = U_{11}$	16	26.4	1	0.435					0.435			

□ indicates the current $\min_{n \text{ available}} U_{nj}$

● the lower bound of the unit whose status just changed.

We now undergo a transition to state $B_1 = \{0, 0, 1\}$, causing unit 1 to become busy. But unit 1 had zero probability of being dispatched in state $\{0, 0, 0\}$ since $L_{11} > U_{31}$, and thus all of the computations remain unchanged. Following the backward regenerative unit-step hypercube tour, we next undergo a transition to state $B_3 = \{0, 1, 1\}$, causing unit 2 to become busy. This requires $(U_{31} - L_{21} + 1)_+ = 8 - 5 + 1 = 4$ values of $q_{nr(e)\|1k}$ to be recomputed; the new computations are easy, since unit 3 is the only unit available. The next transition to state $B_2 = \{0, 1, 0\}$ requires no new computation since $L_{11} > U_{31}$. The next transition is to state $B_6 = \{1, 1, 0\}$, causing unit 3 to become busy. Then the only nonzero $q_{nr(e)\|1k}$'s are $q_{3r(e)\|1k}$ which equal the corresponding $l_{3r(e)}$. The next transition to B_7 causes all units to become busy and the probability $q_{nr(e)\|1k}$ is zero for all n. (Recall that with a zero-line capacity system, calls that arrive when all N units are busy are lost or handled by some back-up service system.) The final two states B_5 and B_4 require only that unit 2 be dispatched, thus completing our example.

5. PERFORMANCE MEASURES

Once the upward transition rates are generated, the steady state equations of balance are solved in the usual way[1], yielding numerical values for the steady state probabilities $P\{B_k\}$, $k = 0, 1, \ldots, 2^N - 1$. These probabilities allow us to obtain numerical values for the performance measures of interest: *region-wide*: mean travel time, dispatch error probability, workload imbalance, and fraction of dispatches that are inter-district dispatches; *response unit specific*: workload (measured in fraction of time busy servicing calls), mean travel time, fraction of responses of each response unit that are interdistrict; *district specific*: fraction of responses into each district that are interdistrict, mean travel time; *point specific*: mean travel time, dispatch error probability, frequency of patrol passings (in the case of mobile police patrol), fraction of calls handled by response unit n, $n = 1, 2, \ldots, N$. This mixture of performance measures allows one to focus simultaneously on several region-wide objectives while assuring that spatial inequities in the delivery of service are maintained at an acceptable minimum. The dispatch error probability in conjunction with mean travel times and interdistrict dispatch frequencies allows one to compare the advantages and disadvantages of AVL dispatching to any fixed preference policy; such a comparison would require two executions of the model—one for each policy.

Individual workloads

Knowledge of the $P\{B_k\}$'s is sufficient to compute the workload ρ_n of each unit n,

$$\rho_n = \sum_{\{i : b_n = 1\}} P\{B_i\}.$$

The unit-step tour has certain periodicity properties that make computation of the sum particularly easy[9].

Retouring the hypercube

For the remaining performance measures it is necessary to reconstruct the "fine structure" on the hypercube, which requires $q_{na|jk}$ and its sum over a, $P_{n|jk}$. Due to the enormous storage requirements implied by storing these quantities, the decision is made simply to *retour* the hypercube using the computed $P\{B_k\}$ at each step to generate many of the performance measures iteratively. The fact that two tours of the hypercube are made—one to set up the equations of balance and, after their solution, one to compute performance measures—intensifies our interest in assuring that the tour is structured efficiently.

Unit-specific mean travel times

With a fixed preference dispatching strategy, the mean travel time t_{ni} for unit n to reach atom j can be determined simply by knowing the dispatching strategy, the mean travel time from atom i to atom j τ_{ij}, and the time-average statistical locations $L = (l_{nj})$ of the units while available within the region R. However, with AVL dispatching the mean travel time is a complicated function of the real-time location of all available response units, whose availabili-

ties are determined by the system state B_k. Thus the mean travel times are computed iteratively in the second tour of the hypercube.

We use $q_{na|jk}$ to compute the expected travel times, which depend on the location of the unit when dispatched. Define

$t_{n|jk}$ = Expected travel time for unit n, given the incident is in atom j, the state of the system is B_k, and unit n is dispatched.

Thus $t_{n|jk}$ is the expected travel time for unit n (conditioned on atom j and state B_k), given that it is the closest available unit. Recalling that τ_{aj} is the travel time from atom a to atom j, we have for all $P_{n|jk} > 0$,

$$t_{n|jk} = \sum_{a=1}^{J} q_{na|jk}\tau_{aj} / P_{n|jk} \qquad n = 1, 2, \ldots, N. \tag{10}$$

(If $P_{n|jk} = 0$, then $t_{n|jk}$ is undefined.)

Now the travel time t_{nj}, not conditioned on system state, is calculated as follows: For each state $B_k = \{b_N, b_{N-1}, \ldots, b_i, \ldots, b_1\}$ in the second hypercube tour such that $b_n = 0$ and $P_{n|jk} > 0$, t_{nk} is incremented by $t_{n|jk}P_{n|jk}P\{B_k\}$. In other words,

$$t_{nj} \leftarrow t_{nj} + t_{n|jk}P_{n|jk}P\{B_k\}. \tag{11a}$$

After the computations are completed for all states B_k the result is scaled:

$$t_{nj} \leftarrow t_{nj} \Big/ \sum_{\{B_k : b_n = 0\}} P_{n|jk}P\{B_k\}. \tag{11b}$$

Inter-area dispatch frequencies

For the remaining performance measures it is necessary to compute

$$\rho_{nj} = \text{fraction of all dispatches that send unit } n \text{ to atom } j \left(\sum_{n,j} \rho_{nj} = 1\right).$$

Following [1] and the arguments above for travel time,

$$\rho_{nj} = \frac{\sum_{\{B_k : b_n = 0\}} P_{n|jk}P\{B_k\}}{1 - P\{B_{2^N-1}\}} \qquad \text{all } n, j. \tag{12}$$

This too is computed iteratively at each hypercube vertex.

Once ρ_{nj} and t_{nj} are computed for all (n, j), then one uses equations (14)–(20) given in [1] for additional performance measures involving interdistrict dispatch frequencies and mean travel times.

Dispatch error probability

One may wish to execute the hypercube model with a fixed preference dispatching policy in order to compute dispatch error probabilities. This requires the computation during the second hypercube tour of the state-dependent dispatch probabilities for both the fixed preference policy and the AVL policy.

Define for any fixed preference policy

$P_j\{C'\} \equiv$ Prob{dispatching a response unit other than the closest available unit to atom j}.

η_{kj} = total number of response units that are judged optimal under fixed preference dispatching, given state B_k and a call in atom j (this is the number of tied units).

r_m = the number of the mth optimal unit, where $r_m = 1, 2, \ldots, N$ and $m = 1, 2, \ldots, \eta_{kj}$.

If $\eta_{kj} > 1$ it is assumed that the unit dispatched to atom j (given state B_k) is to be chosen randomly from among the tied units $r_1, r_2, \ldots, r_{n_{kj}}$.

Since $(1 - P_{r_m|jk})$ is the probability that some unit other than r_m is the closest one to dispatch, given atom j and state B_k, the unconditional dispatch error probability is given by

$$P_j\{C'\} = \sum_{B_k} \sum_{\substack{r_m \\ m=1,2,\ldots,\eta_{kj}}} (1 - P_{r_m|jk}) \frac{1}{\eta_{kj}} P\{B_k\}, \tag{13}$$

where $P\{B_k\}$ in this case refers to the hypercube model with fixed preference dispatch policy.

Multiple values of system-wide workload

When one desires to execute the hypercube model with several different values of system-wide average workload $\rho = \lambda/N$ (everything else remaining constant), a significant computational efficiency is experienced. This occurs for either AVL or fixed preference dispatching. The efficiency derives from the fact that, regardless of the different number of workload levels desired, *only two tours of the hypercube are required.*

The first tour is again to set up the equations of balance. After they are solved for one value of ρ, they can be altered easily to adjust for the next level since all the upward transition rates (λ_{ij} such that $d_{ij}^{+} = 1$) are changed by the same relative amount. The fine-grain state-dependent dispatching policies remain unchanged and thus there is no need to recompute the λ_{ij}'s by touring the hypercube. The computed steady state probabilities must be stored for each different workload level for use in the second hypercube tour.

The second tour is used just as before to compute iteratively most of the system performance measures. Now at each state the measures for *each* of the workload levels are updated, thereby eliminating the need for more than one additional tour of the hypercube. For large problems, however, there exists a tradeoff between execution time (implying only one additional hypercube tour) and storage (which can grow very large if the performance measures for all workload levels are computed—and stored—simultaneously). In anticipating large problems and limited memory computers, the current programmed version of the model allows the user to choose either option: (1) execute one (*re*) tour of the hypercube to compute performance measures for *each* workload level specified; (2) execute only one (*re*) tour of the hypercube, computing performance measures for all workload levels simultaneously.

6. NINE-UNIT EXAMPLE

In this section we apply AVL dispatching to the $N = 9$ unit square region shown in Fig. 2. Here each unit, when available, is equally likely to be located in one of 16 evenly spaced atoms defining its district. Calls for service are uniformly distributed over all $9 \times 16 = 144$ atoms. Calls that arrive when all nine units are busy are entered in a queue of calls that is depleted on a first-come, first-served basis. Travel times are directly proportional to right-angle distance.

We wish to compare AVL with two popular methods of fixed preference dispatching: Strict Center of Mass (SCM) and Modified Center of Mass (MCM). A similar analysis was reported previously ([4], Chaps 6, 7) using a Monte Carlo simulation model.

Suppose atom j is located at (x_j, y_j). To specify the two fixed preference policies, we define the *center of mass* of unit n (x_n^0, y_n^0), where

$$x_n^0 = \sum_{j=1}^{J} l_{nj} x_j$$

$$y_n^0 = \sum_{j=1}^{J} l_{nj} y_j$$

and the center of mass of calls in district n,

$$\bar{x}_n = \sum_{j \in \text{district } n} f_j x_j \Big/ \sum_{j \in \text{district } n} f_j$$

$$\bar{y}_n = \sum_{j \in \text{district } n} f_j y_j \Big/ \sum_{j \in \text{district } n} f_j.$$

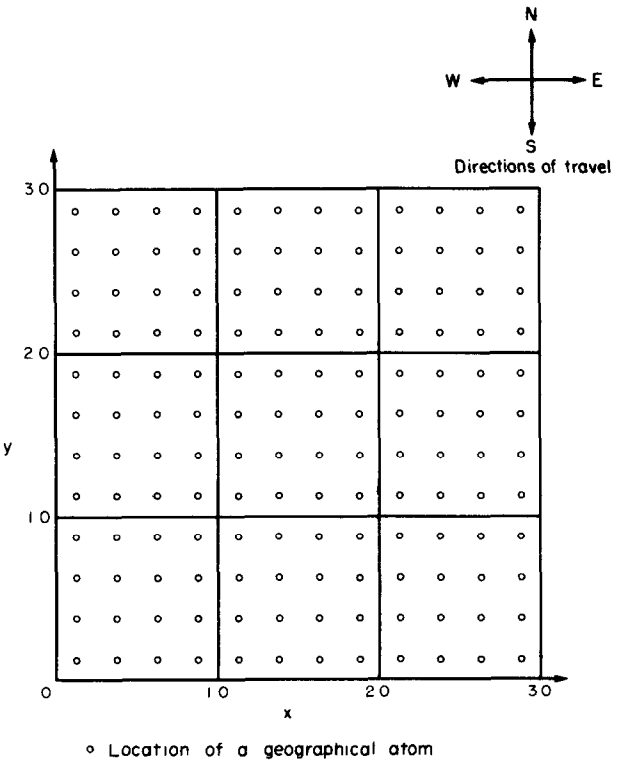

Fig. 2. Nine-unit square region.

Now, an SCM dispatching strategy derives its vector of fixed preferences for units for calls from atom $j \in$ district m by rank ordering according to travel distance from each unit's center of mass to the call's center of mass:

Unit n is the ith preferred if $|x_n^0 - \bar{x}_m| + |y_n^0 - \bar{y}_m|$ is the ith smallest travel distance.

A more sophisticated strategy, the MCM strategy, derives its vector by rank ordering according to travel distance from each unit's center of mass to the atom containing the call:

Unit n is ith preferred if $|x_n^0 - x_j| + |y_n^0 - y_j|$ is the ith smallest travel distance.

The MCM strategy includes information on the exact location of the call whereas the SCM strategy does not. Thus, we would expect that mean travel times would be highest for SCM, second highest for MCM, and lowest for AVL.

Nine runs of the hypercube model were performed for each strategy: SCM, MCM and AVL, each run representing a different level of workload ρ. Here $\rho \equiv \lambda/9\mu = \lambda/9$; it is important to note that since there are no lost calls ρ is identically equal to the average utilization factor (fraction of time busy) for units in the system. When each run was executed separately, the average cost per run using the M.I.T. IBM 370/168 computer was about \$4.00 for the fixed preference strategies, \$12.00 for the AVL strategy and \$15.00 for both the AVL strategy and dispatch error probabilities with a non-AVL strategy (both AVL and non-AVL strategies being computed at the same time.) When four or more runs were executed together (implying only two tours of the hypercube), cost savings in excess of fifty percent were achieved.

Mean travel time reduction vs ρ

At a given value of ρ the mean travel time reduction caused by AVL dispatching is simply the difference between the region-wide travel time of the fixed preference strategy and that of the AVL strategy. Curves displaying mean travel time reductions for both SCM and MCM strategies are shown in Fig. 3. As one can see, the greatest travel time savings available from AVL occur for small values of ρ, when most units are available and the number of possible dispatching choices is large. For MCM dispatching, the mean travel time reduction decreases

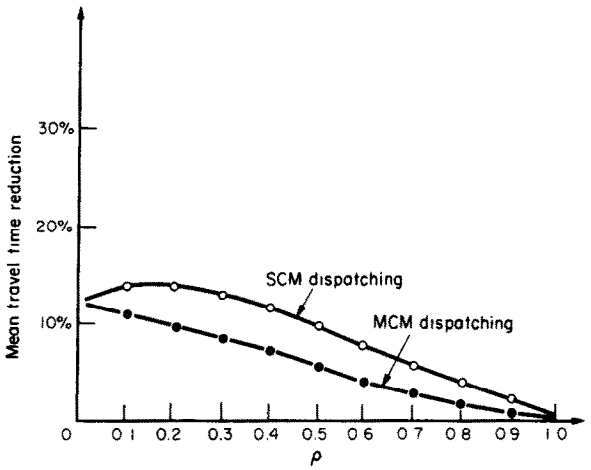

Fig. 3. Mean travel time reduction vs ρ, nine-unit square region.

monotonically with ρ whereas that for SCM dispatching is a unimodel function of ρ, reaching a maximum at about $\rho = 0.20$. This unimodality is caused by the relatively high conditional dispatch error probability associated with a dispatch decision at small ρ, given that the district unit is busy (but because of small ρ, most near-by units are available); this high conditional error probability is caused by ignoring the call's exact location, thereby often yielding ties in dispatch preferences where it is clear that one unit is more likely to be closer than another. Note that for both SCM and MCM the comparative advantage of AVL dispatching converges to zero as ρ converges to 1.0, a point at which there is virtually no choice allowed in selecting an available unit for dispatch.

Probability of dispatch error vs ρ

Similar curves of probability of dispatch error vs ρ are given in Fig. 4. Again, dispatch error tends to be high when ρ is small. Dispatch error is a monotonically decreasing function of ρ for MCM dispatching and a unimodal function for SCM dispatching. Note that at $\rho = 0.0$, for instance, approximately 28% of dispatch decisions could be improved (in the sense of smaller travel times) by employing AVL dispatching compared to either of the fixed preference dispatching strategies.

Amount of interdistrict dispatching vs ρ

AVL dispatching has its disadvantages as well as its advantages. One disadvantage is that the amount of interdistrict dispatching increases, thereby decreasing the amount of contact a

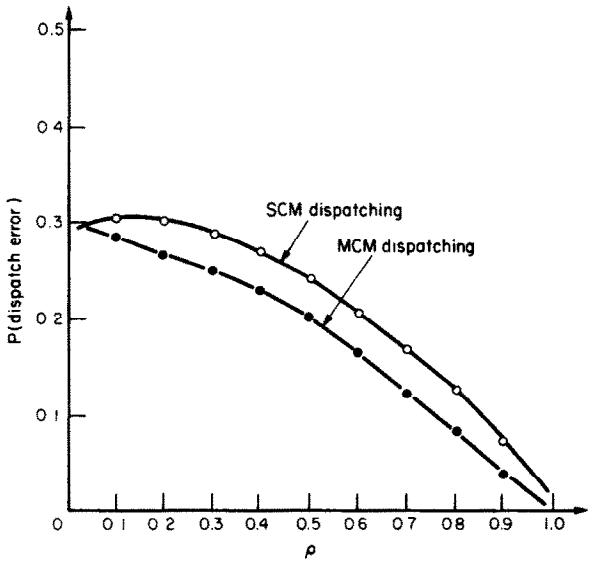

Fig. 4. Probability of dispatch error vs ρ, nine-unit square region.

server has with citizens in his "own" district. In police applications, where "district" is a police beat or sector, this is said to cause a loss of beat or sector "identity."

Curves indicating the extent of interdistrict dispatching for each of the three strategies are shown in Fig. 5. As expected for SCM and MCM strategies, the fraction F_I of dispatches that are interdistrict dispatches almost equals ρ for small and moderate values of ρ. This is intuitively explained by the fact that a fraction ρ of calls for service find the district unit busy and thus require an out-of-district unit if immediate service is to be delivered. Also as expected, the fraction F_I is significantly increased for low and moderate values of ρ by switching to AVL dispatching. For instance, F_I at $\rho \approx 0.0$ is about 0.28, which is just what we expect since the dispatch error probability at $\rho \approx 0.0$ is about 0.28, thereby necessitating interdistrict dispatches for 28% of calls for service even when all units are available.

Probability of dispatch error as a function of incident location

Each of the previous three sets of curves had been derived earlier with a simulation model [[4], Chaps. 6, 7]. However, the results reported here do not suffer from sampling error and they were generated at a cost at least an order of magnitude less than the simulation costs. Moreover, the hypercube model produces point-specific as well as area-averaged performance measures, something that is too expensive to obtain from a simulation model.

An important consideration in the case of AVL dispatching, the point-specific dispatch error probability has heretofore been unobtainable. Computations of this quantity with the hypercube model have indicated that region-wide dispatch error probability can be rather modest, say 12%, yet specific atoms can have an associated dispatch error probability as high as 50% or more. This significant degree of spatial inhomogeneity in dispatch error probability is illustrated in Fig. 6, which displays for a fixed ρ ($\rho = 0.1$) this probability as a function of incident location x as one travels west to east at a y value fixed at $y = 1.375$. As can be seen from the figure, dispatch error probability increases sharply near the boundaries of districts, reflecting the fact that a unit on the other side of the boundary has a significant probability of being closest to the call. Thus AVL dispatching would yield the sharpest reduction in travel times to incidents located near the borders of other districts.

7. DISCUSSION

The methods reported here for efficiently implementing AVL dispatching within the hypercube framework apply to a more general class of queuing system. In queuing terms, there are J classes of customers (corresponding to the J atoms), each class j arriving as an independent Poisson process to an $M/M/N$ queue with distinguishable servers. Server n, if available, will be in posture i with probability l_{ni} $\left(\sum_i l_{ni} = 1\right)$. And there is a cost τ_{ij} of assigning a

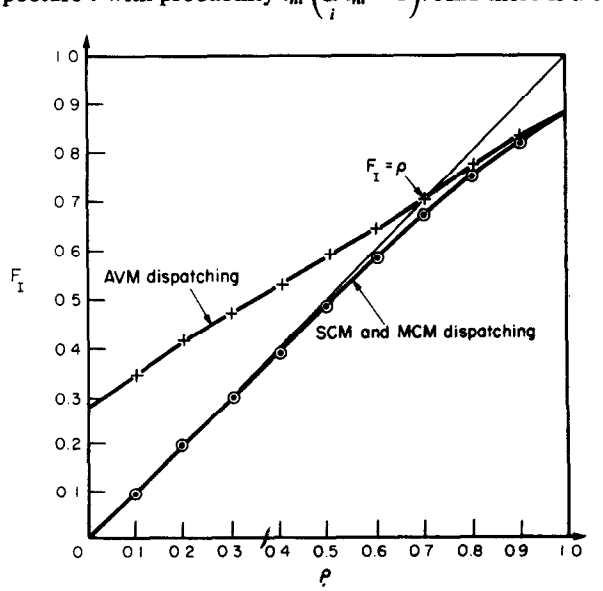

Fig. 5. Fraction of dispatches that are interdistrict vs ρ, nine-unit square region.

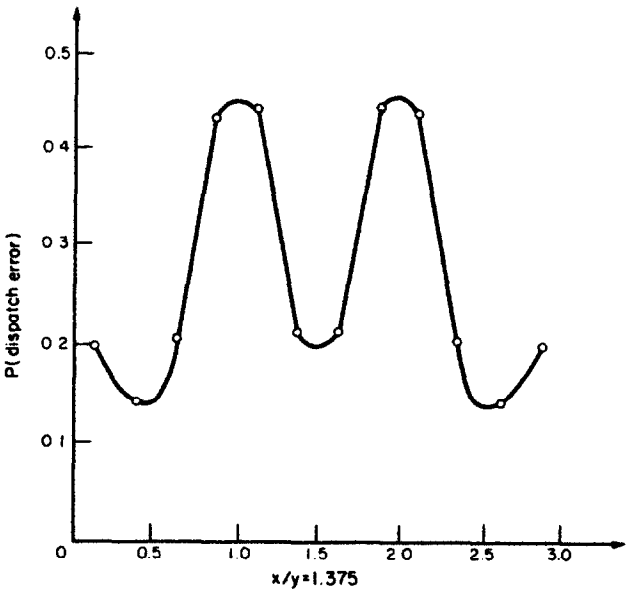

Fig. 6. Probability of dispatch error as a function of location of incident, nine-unit square region.

server in posture i to a class j customer. The methods discussed in this paper allow one to model analytically (and solve numerically) the behavior of this system, operating under a policy that always assigns that server with minimum (immediate) cost. It should be noted, that in the spirit of the models of Carter, Chaiken and Ignall[8] and of Jarvis[9], that a server assignment policy that minimizes immediate cost does not in general minimize time-average cost of the system (although the difference between the two policies in system-wide mean cost has been found to be small—less than one percent when the cost has been travel time). In nonspatial applications one can imagine that customer classes could be defined by one or more of the following: yearly income, educational level, amount of cash carried, ethnicity, physical size, sex, or (in computer applications) type or source of data message. Too, a server's posture could be specified by any number of variables, including priority of his current activity (which is viewed as preemptable), degree of fatigue, amount of cash on hand, amount of core storage available.

Acknowledgement—The work reported here was supported by the National Science Foundation, Division of Advanced Productivity, Research and Technology, Grant No. GI38004. All computations were performed at the M.I.T. Information Processing Center.

REFERENCE

1. R. C. Larson, A hypercube queuing model for facility location and redistricting in urban emergency services, *Comput. Ops Res.* 1, 67 (1974).
2. M. Bellmore, Automatic car locators, pp. 149–151 of appendix E (electronics equipment associated with the police car) in President's Commission on law enforcement and administration of justice, *Task Force Rep. Science and Technology*, U.S. Government Printing Office, Washington, D.C. (1967).
3. R. A. Bales, A police car simulation model: conventional versus AVM dispatching, in *Proc. 1970 Carnahan Conf. on Electronic Crime Countermeasures*, University of Kentucky and Institute of Electrical and Electronic Engineers, pp. 1–23. 16–18 April (1970).
4. R. C. Larson, *Urban Police Patrol Analysis*. The MIT Press, Cambridge, Mass. (1972).
5. W. C. Scales, (guest editor), Special issue on automatic vehicle monitoring, *IEEE Trans. Veh. Tech.* 26, (1977).
6. G. R. Hansen and W. G. LeFlang, *Application of Automatic Vehicle Location in Law Enforcement*. Jet Propulsion Laboratory, document no. JPL 5040-17, Pasadena, California (1976).
7. J. P. Jarvis, Optimization in stochastic service systems with distinguishable servers, *tech. rept. TR*-19-75. Massachusetts Institute of Technology, Operations Research Center, Cambridge, MA, June (1975).
8. R. A. Howard, *Dynamic Probabilistic Systems, Volume I: Markov Models*. John Wiley, New York (1971).
9. R. C. Larson, *Some Useful Properties of a Backward Regenerative Unit-Step Tour of the Hypercube*, to appear.
10. G. M. Carter, J. M. Chaiken and E. Ignall, Response areas for two emergency units, *Ops Res.* 20, 571 (1972).
11. J. P. Jarvis, Optimal dispatch policies for urban server systems, *tech. rept. TR*-02-73. Massachusetts Institute of Technology, Operations Research Center, Cambridge, MA, September (1973).
12. Public Systems Evaluation, Inc., *Evaluating an Implemented AVL System*. Available through the Office of Evaluation, National Institute of Law Enforcement and Criminal Justice, Law Enforcement Assistance Administration, U.S. Department of Justice, Washington, D.C. (1976).

13. R. C. Larson, Computer program for calculating the performance of urban emergency service systems: user's manual (batch processing) program version 75-001 (batch), *tech. rept. TR*-14-75. Massachusetts Institute of Technology, Operations Research Center, Cambridge, MA, 02139, March (1975).
14. R. C. Larson, K. W. Colton and G. C. Larson, Evaluating a police-implemented AVM system: The St. Louis experience (Phase I), *IEEE Trans. Veh. Tech.* **26**, pp. 60–70 (1977).

APPENDIX:* NUMBER OF PROBABILITY COMPUTATIONS REQUIRED WITH THE BACKWARD REGENERATIVE UNIT-STEP TOUR

The purpose of this appendix is to derive equation (9), which specifies the number of computations of $q_{na|jk}$ required for a particular atom j in the course of touring the hypercube in a unit-step fashion (according to the backward regenerative tour), with units rank-ordered according to the L_{ij}'s:

$$L_{Nj} < L_{(N-1)j} < \cdots < L_{1j}$$

In the derivation we also require

$$U_{Nj} < U_{(N-1)j} < \cdots < U_{1j}$$

We motivate the general result from the $N = 4$ unit problem which is summarized in Table A-1, whose columns contain (left to right) the unit number, the number of state changes (or transitions) during a tour, the number of computations of $q_{na|jk}$ per transition, and the transition(s) at which that number of computations occur. Table A-2, summarizing the hypercube tour, contains the binary representations of the states in the tour, and a simple linear index denoting each state. We focus on state transitions and the number of computations required per transition. Since the computations of $q_{na|jk}$ must start at state S_1 (0000), we include "going to the start state" as a transition. Thus, there are 2^N transitions for an N unit problem, with unit n experiencing 2^{N-n} transitions (excepting unit N which experiences $2^0 + 1 = 2$ transitions).

We partition transitions into two classes: (1) those which involve a unit n for which L_{nj} is the minimum of all L_{ij}'s currently in the available set $\mathcal{A}_k$; (2) all other transitions. For those in class 2, there is always at least one available unit that can be closer to atom j than unit n, regardless of the location of unit n; for those in class 1, unit n can be closest, regardless of the locations of the other available units.

Table A-1. Summary of computational work required for an $N = 4$ unit problem

Unit number	Number of state changes during tour	Number of computations per transition	Transition at which computations occur
4	2	$U_4 - L_4 + 1$	start up at state 1
		$U_2 - L_2 + 1$	8→9
3	2	$(U_4 - L_3 + 1)_+$	4↛5
		$U_3 - L_3 + 1$	12→13
2	4	$(U_4 - L_2 + 1)_+$	2→3, 6→7
		$(U_3 - L_2 + 1)_+$	14→15
		0	10→11
1	8	$(U_4 - L_1 + 1)_+$	1→2, 3→4, 5→6, 7→8
		$(U_3 - L_1 + 1)_+$	13→14, 15→16
		$(U_2 - L_1 + 1)_+$	9→10
		$U_1 - L_1 + 1$	11→12

Table A-2. Unit-step tour of four-dimensional hypercube

State S_k of unit-step tour	Index number k of state S_k
0000	1
0001	2
0011	3
0010	4
0110	5
0111	6
0101	7
0100	8
1100	9
1101	10
1111	11
1110	12
1010	13
1011	14
1001	15
1000	16

*While this Appendix is self-contained, certain mathematical properties of the backward regenerative unit-step tour are invoked. Those wishing additional background in this area may wish to consult [9].

Consider first class 2 transitions. The transition from state S_1 to S_2 is of this type. Here unit 1 becomes busy while the other 3 units remain available. If there is any overlap in the table of $q_{nr(e)ijk}$ values, it occurs between the upper bound of unit 4 and the lower bound of unit 1. The amount of overlap is $U_4 - L_1 + 1$, if positive. Since the overlap is zero otherwise (implying 0 computations resulting from the 1-to-2 transition), we abbreviate the number of computations as $(U_4 - L_1 + 1)_+$, where $(x)_+ = x$ only if $x > 0$; otherwise $(x)_+ = 0$. Since there are $2^{N-1-1}(=2^2 = 4)$ transitions involving unit 1 while unit 4 is available, $(U_4 - L_1 + 1)_+$ computations are experienced 4 times. Similarly there are two transitions in which unit 1 changes state and unit 3 is available (while unit 4 is busy); each of these requires $(U_3 - L_1 + 1)_+$ computations. Once unit 1 makes a transition when units 4 and 3 are busy and unit 2 is available, requiring $(U_2 - L_1 + 1)_+$ computations. In general for an N unit problem, class 2 transitions of unit 1 require

$$\sum_{n=2}^{N} (U_n - L_1 + 1)_+ 2^{n-1-1}$$

computations. The same reasoning applied to class 2 transitions of an arbitrary unit m yields for the number of computations

$$\sum_{n=m+1}^{N} (U_n - L_m + 1)_+ 2^{n-m-1}.$$

Summing this last quantity over all m, we obtain the first term on the right hand side of equation (9).

Now consider class 1 transitions and subdivide these into two subclasses: (a) those which result in the unit m becoming available; (b) those which result in unit m becoming busy. For an N unit problem, exactly one transition of type (1b) will occur for each unit N, N-2, N-4, ..., and exactly one transition of type (1a) will occur for each unit N-1, N-3, N-5, ... A (1b) type transition must occur for unit N in transitioning from state $S_{2^{N-1}}$ to $S_{(2^{N-1}+1)}$, by construction of the tour which adds a "one" in position N at state $S_{(2^{N-1}+1)}$. Similarly such transitions must occur for all units m resulting from subtracting an even number from N, since at points in the tour which begin with the $2n$ highest indexed units busy (n integer), the tour for units 1, 2, ..., $(N-2n)$ has reversed an even number of times, yielding forward "subtours." The length of each subtour is equal to the number of N-tour entries for which the $2n$ highest indexed units are busy. If the Nth digit switches to one only once in an N-digit tour, the nth digit must switch to one only once in an n-digit forward subtour. If units N, N-2, N-4, ... incur type (1b) transitions for an N-unit problem, the reversal of the transition sequence when unit $N+1$ is busy in an $(N+1)$-unit problem indicates that those same units will experience type (1a) transitions in such a problem. Thus, units m obtained by subtracting an odd number from N each experience a type (1a) transition.

For transitions of type 1(a) (resulting in unit m becoming the possibly closest unit), it is clear that the number of computations of $q_{na|ijk}$ required is $U_m - L_m + 1$. For transitions of type 1(b), unit m is no longer available. But, by the construction of the backward regenerative unit-step tour, unit $m-1$ must also be unavailable (since the transition from 0 to 1 for unit m follows 2^{m-2} consecutive states in which unit $m-1$ is busy). Also by the construction of the tour, unit m-3 must be free when unit m makes its type 1(a) transition. This is so since the tour has stepped backward to state S_1, picking up all zeroes except for unit m-1 at the point of unit m's transition to busy status. Thus, if we do not count erasing zeroes in positions $L_3 - L_1$ as computations, then the number of computations required as a result of unit m making a type 1(b) transition is $U_{m-2} - L_{m-2} + 1$ (since unit m-2 is now the possibly closest unit).

In the $N = 4$ example, the transition 8→9 is of type 1(b), causing unit 4 to become busy and requiring $U_2 - L_2 + 1$ computations. Unit 2's transition 10→11 is also of this type, yielding the state 1111, implying that all units are busy; here no ordinary computations are required and our formula is still valid since $U_{m-2} - L_{m-2} + 1$ is not defined for U_0 and L_0.

Finally, since we include the starting state as a transition, we must add $U_N - L_N + 1$ as the number of computations required to start the process. Adding all computations required for type (1a) and (1b) transitions and for the starting state, we obtain the second term on the right hand side of equation (9), and our derivation is complete.